Historique des travaux autour du problème P||Cmax

**Introduction**

**Présentation du problème.**

**Parallélisme.**

Le parallélisme est un type d'architecture informatique dans lequel plusieurs processeurs exécutent ou traitent une application ou un calcul simultanément. IL aide à effectuer de grands calculs en divisant la charge de travail entre plusieurs processeurs, qui fonctionnent tous en même temps.

Il existe quatre types de parallélismes, définis par la taxonomie de Flynn(1). Cette classification est basée sur deux notions : le flot d’instructions (simple ou multiple), et le flot de donnée (simple ou multiples) ; un algorithme est un flot d’instructions à exécuter sur un flot de données.

|  |  |  |
| --- | --- | --- |
| Instructions / Données | Simple | Multiple |
| Simple | **SISD**  premiers PC  machine de Von Neumann  Obsolète, car tous les PC sont désormais multi-cœur. | **SIMD**  Machines synchrones  Pipeline  Exécution d’une instruction unique sur des données différentes. |
| Multiple | **MISD**  Machines vectoriels  Tableau de processeurs  Exécute plusieurs instructions sur une même donnée. | **MIMD**  Multi processeurs à mémoire distribuée.  Multi processeurs à mémoire partagée (multi-cœur).  Multi Ordinateur |

Taxonomie de Flynn

Les premières machines parallèles étaient des réseaux d’ordinateurs, et des machines vectorielles (faiblement parallèles, très coûteuses), telles que l’IBM 360, les Cray1. La plupart des machines parallèle contemporaines sont désormais MIMD.

On peut définir une machine parallèle comme un ensemble de processeurs qui coopèrent et communiquent.



IBM 360-91 (le plus rapide et le plus puissant en service en 1968) NASA. Centre de vols de Greenbelt (Md)

**Ordonnancement**

Sur une machine non parallèle, les tâches sont exécutées séquentiellement, les unes après les autres. Certaines tâches, ou jobs peuvent demander plus de temps que d’autres pour être entièrement traitées.

Lorsque plusieurs ressources (processeurs, machines, cœurs) sont disponibles, se pose alors, un problème d’ordonnancement.

Celui-ci consiste à organiser, dans le temps, les jobs à exécuter, en les affectant à une ressource donnée, de manière à satisfaire un certain nombre de contraintes, tout en optimisant un ou des objectifs.

L’ordonnancement, fait partie de la catégorie des problèmes d’optimisation combinatoire.

Les problèmes qui s’y rattachent sont très variés.

Premièrement, la nature des machines parallèles doit être considérée. Celles-ci peuvent être identiques (Le même temps de traitement sera nécessaire, d’une machine à l’autre) ; uniformes (un quotient de vitesse qi propre à une machine est à appliquer pour chaque tâche affectée à cette machine pour déterminer le temps de traitement nécessaire) ; ou indépendantes (les temps de traitements des tâches sont ni uniformes ni proportionnels d’une machine à l’autre).

Ensuite, des contraintes peuvent affecter les jobs eux-mêmes. Dans le cas d’un problème préemptif, les taches peuvent être interrompues, et reprises ultérieurement. Il est possible que les jobs soient indépendants, ou au contraire, être liées par des relations de précédence. Ces jobs ne sont disponibles qu’à partir d’une certaine date. Ou encore, être de durée égale, ou tous de durée différente.

Pour finir, l’objectif de l’ordonnancement est d’optimiser un critère. Par exemple, minimiser la somme des dates de fin, la somme des retards, le nombre de tâches en retard, ou simplement, le retard total. Mais le plus habituel, est de chercher à minimiser le temps total de traitement de tous les jobs, i.e minimiser le makespan.

**Enoncé du P||Cmax**

Ces diverses possibilités définissent divers problèmes d’ordonnancements différents, recensés et classifiés par Graham et al. [1], qui introduit la notation trois-champs α|β|γ .

Le problème Pm||Cmax se définit ainsi :

**α = α1 α2, détermine l’environnement machines.**

α = P : Les machines sont parallèles et identiques : Un job, une tâche prendra le même temps de traitement qu’il soit exécuté sur une machine ou une autre. Le nombre de machine (m) est variable.

**β c { β1, β2, β3, β4, β5, β6}, détermine les caractéristiques des jobs, ou des tâches.**

β est vide. Ce qui signifie que la préemption n’est pas autorisée (les jobs doivent être exécutés d’une traite, sans interruption ni coupure) et qu’il n’y a pas de relation entre les jobs (ils sont indépendants).

**γ détermine le critère à optimiser.**

γ = Cmax : on cherche à optimiser le makespan, le temps de traitement total.

En d’autres termes, Pm||Cmax consiste à planifier un ensemble N = {1,2,…,n) de n jobs simultanés, pour être traités par m machines identiques et parallèles. Chaque job, qui requière une opération, peut être traité par une des m machines. Le temps de traitement de chaque job (Pi avec i ∈ N) est connu à l’avance. Un job commencé, et complété sans interruption. Les jobs sont indépendants, et une machine ne peut traiter qu’un seul job à la fois.

Transition

Problème combinatoire

NP-Difficile donc pas d’algorithme qui donne le résultat en temps raisonnable

Trouver un algorithme qui donne un résultat le plus proche en un temps

**Résoudre le problème**

* Programmation linéaire
* Algorithme « an LP-Based Algorithm » Mokotoff 1999
* Approximation
* Principe PTAS (programmation dynamique, approximation ε …
* Algorithme « Using dual approximation Algorithm for Scheduling problems : Theorical and Practical Results » (Hochbaum et Shmoys 1987)

* Heuristiques
* Basé LS

Algorithme LPT rule graham 1969

Algorithme LPT-REV (Croce et Scatamacchia, 2018)

Utilisation de la programmation Linéaire pour déterminer les pires-cas.

Déduction d’une heuristique pour éviter les instances des pires cas.

* Basé Bin-Packing

Définition du problème du bin-packing.

Pourquoi peut-il servir à résoudre P||Cmax

Algorithme MULTIFIT

Algorithme COMBINE

Algorithme LISTFIT

* Approche gloutonne

Algorithme SLACK (Croce et Scatamacchia, 2018)

Stratégie gloutonne déduite suite à preuve d’une borne.

* Autres approches
* LDM

Transition

LPT fait toujours référence pour comparer chaque algorithme développé…

..

Les heuristiques, font plus l’objet de recherches ….

**Synthèse**

Récapitulatif complexité / Borne d’approximation… théoriques

Quelques résultats comparés expérimentaux

Avantages inconvénients

**Conclusion**

* Utilisation des algorithmes

Slack est utilisé dans …

* Point de vue personnel
* Recherche documentaire

Document LDM n’explique pas comment les partitions sont construites (semble utiliser LPT)

Difficulté de trouver des applications aux algorithmes

Beaucoup de documents indisponibles, payant…

Acteurs prolifiques Graham, Mokotoff, …

* Futur ? …

Annexes

(1). A été trouvé aussi sous le nom de « taxonomie de Tanenbaum »

Un site effectue régulièrement, un top 500 des machines les plus puissantes :

<https://www.top500.org/>

Des sites internet rassemblent des documents concernant les problèmes d’ordonnancement :

<http://www.mathematik.uni-osnabrueck.de/research/OR/class/>

<http://schedulingzoo.lip6.fr/>

Références

[1]

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Rinnooy Kan, A. H. G. (1979).Optimization and approximation in deterministic sequencing and scheduling: a survey. In P. L. Hammer, E. L. Johnson, & B. H. Korte (Eds.), Discrete optimization II, annals of discrete mathematics (Vol. 5, pp. 287–326).